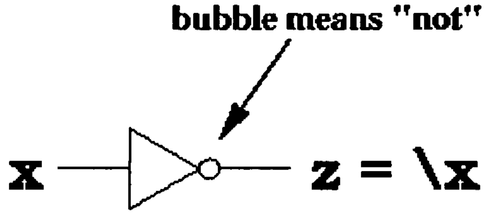
What's a Tri-state Buffer?

Introduction

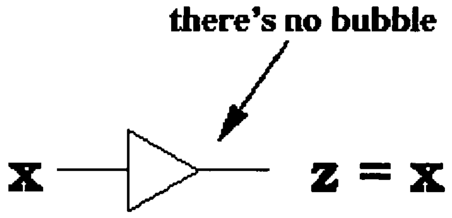
Before we talk about tri-state buffers, let's talk about an *inverter*. You can read about inverters in the notes about Logic Gates. However, we'll repeat it here for completeness.

An *inverter* is called a NOT gate, and it looks like:



The inverter is a triangle, followed by a circle/bubble. That circle sometimes appears by itself, and means negation.

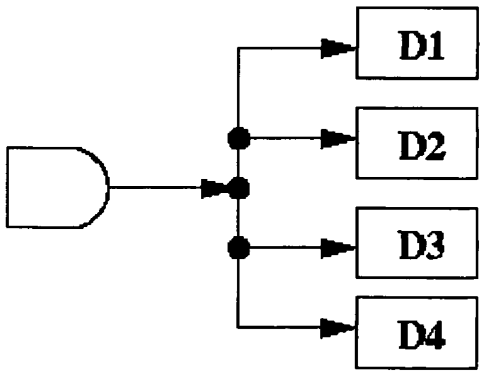
What if we remove the circle? What kind of gate would we have? We'd have a *buffer*.



You might think that a buffer is useless. After all, the output is exactly the same as the input. What's the point of such a gate?

The answer is a practical issue from real circuits. As you may know, logic gates process 0's and 1's. 0's and 1's are really electric current at certain voltages. If there isn't enough current, it's hard to measure the voltage.

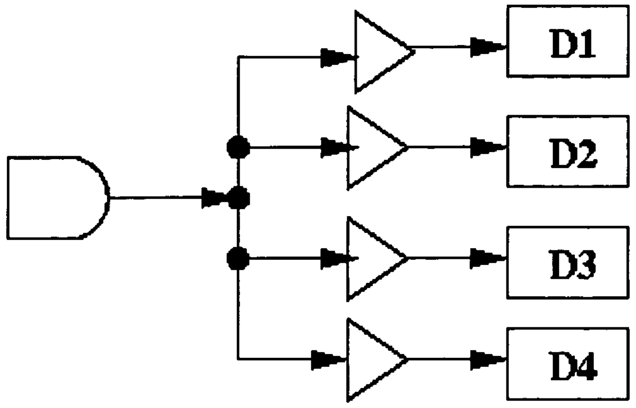
The current can decrease if the fan out is large. Here's an example:



The "fan out" is the number of devices that an output is attached to. Thus, the AND gate above is attached to the inputs of four other devices. It has a fan out of 4.

If the current coming out of the AND gate is **i**, then assuming each of the four devices gets equal current, then each device gets **i / 4** of the current.

However, if we put in a buffer:



then the current can be "boosted" back to the original strength. Thus, a buffer (like all logic gates) is an *active* device. It requires additional inputs to power the gate, and provide it voltage and current.

You might wonder "Do I really need to know this? Isn't this just EE stuff?". That's true, it is. The point of the discussion was to motivate the existence of a plain buffer.

Tri-state buffer: It's a Valve

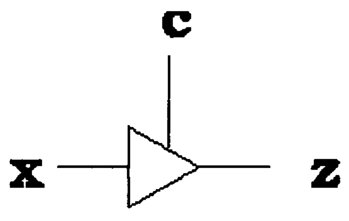
A buffer's output is defined as **z** = **x**. Thus, if the input, **x** is 0, the output, **z** is 0. If the input, **x** is 1, the output, **z** is 1.

It's a common misconception to think that 0 is nothing, while 1 is something. In both cases, they're something. If you read the discussion in What's a Wire, you'll see that a wire either transmits a 0, a 1, or "Z", which is really what's nothing.

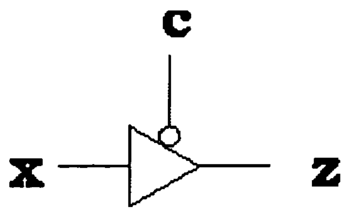
It's useful to think of a wire as a pipe, and 0 as "red kool aid" and 1 as "green kool aid" and "Z" as "no kool aid".

A *tri-state* buffer is a useful device that allows us to control when current passes through the device, and when it doesn't.

Here's two diagrams of the tri-state buffer.



tri–state buffer with active high control



tri–state buffer with active low control

A tri-state buffer has two inputs: a data input **x** and a control input **c**. The control input acts like a valve. When the control input is active, the output is the input. That is, it behaves just like a normal buffer. The "valve" is open.

When the control input is not active, the output is "Z". The "valve" is ~~open~~, and no electrical current flows through.

close

Thus, even if **x** is 0 or 1, that value does not flow through.

Here's a truth table describing the behavior of a active-high tri-state buffer.

c	x	z
0	0	Z
0	1	Z
1	0	0
1	1	1

In this case, when the output is **Z**, that means it's high impedance, neither 0, nor 1, i.e., no current.

As usual,the condensed truth table is more enlightening.

c	z
0	Z
1	x

As you can see, when **c** = **1** the valve is open, and **z** = **x**. When **c** = **0** the valve is closed, and **z** = **Z** (e.g., high impedance/no current).

Active-low tri-state buffers

Some tri-state buffers are active low. In an active-low tri-state buffer, **c** = **0** turns open the valve, while **c** = **1** turns it off.

Here's the condensed truth table for an active-low tri-state buffer.

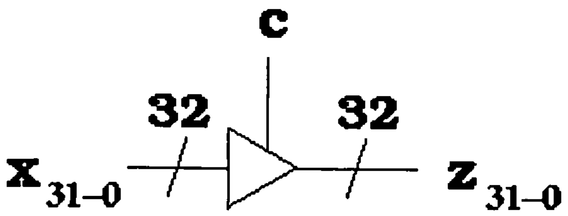
c	z
0	x
1	Z

As you can see, when **c** = **0** the valve is open, and **z** = **x**. When **c** = **1** the valve is closed, and **z** = **Z** (e.g., high impedance/no current). Thus, it has the opposite behavior of a tri-state buffer.

Multi-bit Tri-State Buffers

So far, we've talked about a tri-state buffer controlling the output of a single wire. However, it's more common to deal with many wires.

Here's an example:



tri-state buffer with a bus of 32 bits

In this case, we have 32 wires coming into the tri-state buffer. We have 32 wires coming out of the tri-state buffer.

There's still only 1 control bit.

This can easily be implemented using 32 tri-state buffers taking one bit as input and one bit as output.

Why Tri-State Buffers?

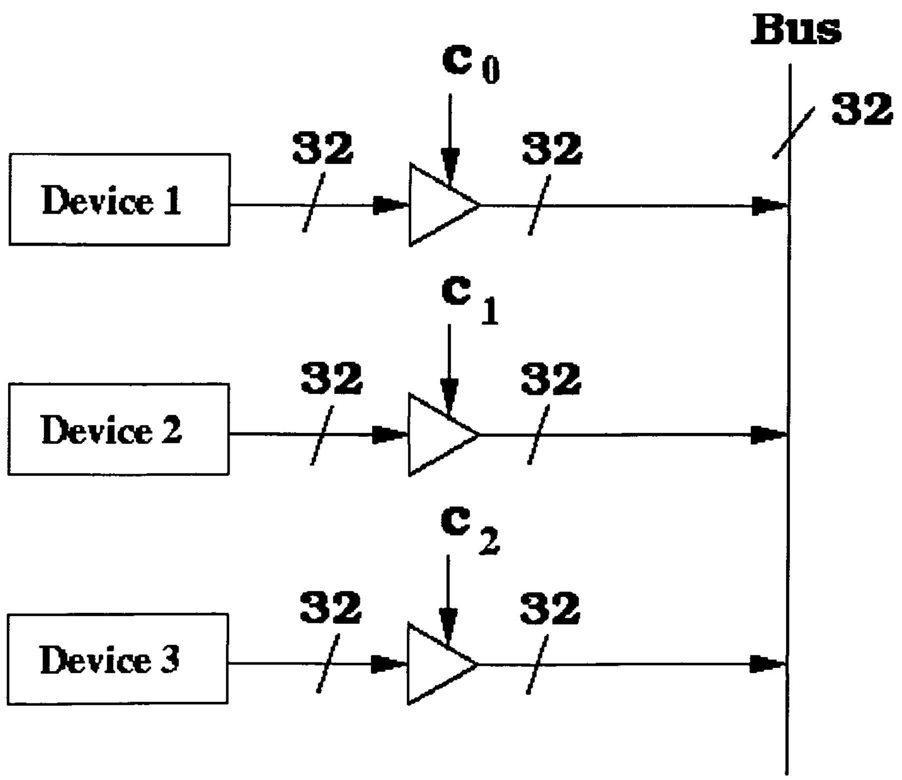
We've had a long discussion about *what* a tri-state buffer is, but not about what such a device is good for.

Recall (from earlier) that a common way for many devices to communicate with one another is on a bus, and that a bus should only have one device writing to it, although it can have many devices reading from it.

Since many devices always produce output (such as registers) and these devices are hooked to a bus, we need a way to control what gets on the bus, and what doesn't.

A tri state buffer is good for that.

Here's an example:



There are three devices, each of which output 32 bits. These devices have their outputs hooked to a 32 bit bus.

We want to prevent more than one device from writing to the bus. Ordinarily, these devices always generate output, so we're in trouble merely by attaching more than one device's output to the bus.

As long as at most one of the following control bits, $\mathbf{c_0}$, $\mathbf{c_1}$, $\mathbf{c_2}$, is 1, the bus is fine. That is, the bus will not have two devices attempting to write to it at the same time.

Alternative: Using a MUX

Tri-state buffers are one way of preventing an output from making it to the bus.

An alternate way is to use a MUX. For example, we might have a 32 bit, 3-1 MUX. The advantage of a MUX is that we're guaranteed only one device makes it to the bus. The drawback is that we might want no devices to make it to the bus.

One solution is to add an **enable** input to a MUX. When the enable is active, the output is selected from one of the inputs. When the enable is not active, then the output is **Z**.

Summary

A tri-state buffer is a device that allows you to control when an output signal makes it to the bus. When the tri-state buffer's control bit is active, the input of the device makes it to the output. This is when the "valve" is open.

When it's not active, the output of the device is **Z**, which is high-impedance or, equivalently, nothing. This is when the "valve" is closed, and no electrical signal is allowed to pass to the output.

The "valve" analogy helps make it easy to understand the behavior of a tri-state buffer.